PET User Notes

Publication of the PET User Group P.O. Box 371 Montgomeryville, PA 18936

Volume 1 Issue 4 May-June 1978

NIM for PET

Jim Butterfield, Toronto

The AND, OR, and NOT relationships can be used for logical tests such as IF A=9 OR B=0 .. They can also be used for manipulation of binary numbers. For example, PRINT 7 AND 10 produces 2. If this isn't clear, you need to brush up on binary numbering and logical operators. In the meantime, you can still play NIM.

The "missing" logical operator, EOR, has been created as FNE. (line 110).

```
 90 PRINT "cs NIM       J. BUTTERFIELD":IQ=.9
100 N=3          set number of piles, e.g. 3
110 DEF FNE(X)=(A(J) OR E) AND (NOT (A(J) AND E))
                                       exclusive OR
120 DIM A(N): INPUT "WANT INSTRUCTIONS";Z$
130 IF LEFT$(Z$,1)="N" GOTO 200
140 ? "PICK FROM";N;"PILES-AS MANY ITEMS AS"
150 ?"YOU LIKE FROM ANY PILE cd THEN I PLAY cd"
160 ?"YOU WIN IF YOU GET THE LAST ITEM cd"
170 ?"GOOD LUCK! cd HIT ANY KEY TO START"
180 GET Z$:IF Z$="" GOTO 180
200 FOR J=1 TO N
205 A(J)=INT(RND(1)*12+5)          generate random piles
210 IF J=1 GOTO 240
220 FOR K=1 TO J-1:IF A(J)=A(K) GOTO 205    avoid duplicates
230 NEXT K
240 NEXT J:M=0
300 T=0:? "cs";           each move starts here
310 FOR J=1 TO N:? CHR$(J+64);"=";A(J):T=T+A(J)
                                  print and total piles
320 NEXT J: IF T=0 GOTO 700       if all gone, game over
330 M=1-M: IF M=0 GOTO 500        PET's move
360 ?"cd TAKE YOUR PICK--"
370 INPUT "PILE";Z$
380 P=ASC(Z$)-64:IF P< 1 OR P> N GOTO 370
390 INPUT "TAKE HOW MANY";Z
400 IF Z> A(P) THEN ? "THERE ARE ONLY";A(P) : GOTO 370
410 IF Z <1 or Z <> INT(Z) THEN ? "WHA?": GOTO 370
                                  illegal quantity
420 A(P) = A(P)-Z : GOTO 300
500 E=0 : F=0
510 FOR J=1 TO N
520 E=FNE(0):IF A(J)> F THEN F=A(J): J1=J  form EOR, all piles
530 NEXT J
540 FOR J=1 TO N
550 R=FNE(0): IF R<=A(J) GOTO 600   find a move
560 NEXT J : STOP                   no moves? something wrong
600 IF R=A(J) OR IQ > RND(1) THEN J=J1:R=A(J)-1  dumb move?
610 ?"I PICK";A(J)-R;"UNITS FROM PILE ";CHR(J+64)
620 A(J)=R: FOR J=1 TO 2000: NEXT J: GOTO 300
                                  pause before next display
700 W$="YOU":IF M=0 THEN W$="I"
710 ?"*** r";W$;" WIN!"
720 IF M=0 GOTO 740
730 IQ=IQ*IQ : ? "I'LL PLAY BETTER"; : GOTO 750
740 IQ=SQR(IQ): ? "I'LL TAKE IT EASIER";
750 ? " NEXT GAME... cd" : INPUT "ANOTHER GAME";Z$
760 IF LEFT$(Z$,1)<>"N" GOTO 200
```

cs=clear screen cd=cursor down r=reverse

[ed. note: NIM demonstrates several aspects of PET BASIC in addition to the logical operators. Jim used CHR$ in line 310 to convert from numeric 1, 2, or 3 to A, B, or C. The reverse conversion ASC was used to translate alphabetic entry 1, 2, or 3 (if N=3). The program also cleverly adjusts its skill by changing the variable IQ (what else?)]

General Notes

Commodore recently began sending a "Revised Introduction to your PET" with PET shipments. The booklet is about the same as the previous version, but does omit the memory map and the edge connector designations. I hope Commodore gets some decent documentation out soon. I know they can do it -- the KIM and 6500 Hardware and Programming manuals are very good.

The PET main circuit board has changed several times since last year. Commodore has been using 2114 RAM's instead of their own 6550 chip, and is apparently going to be an alternate source for the Intel memory IC (the Commodore version will be the 6114, with additional plans to produce a 6316 16K RAM as an alternate source for the Intel 2316).

The Commodore printer (PET Formatting Printer) should be available in late July. Although the specifications are a little sketchy, the unit appears to be an excellent value at $595. Features include 80 column width on 8 1/2 inch paper at 120 characters/second. All PET characters (upper case, lower case, and graphics) will print using a 7 by 8 dot matrix print head. The printer will be assigned physical address 4 on the IEEE-488 bus. The secondary address specified in the OPEN for the device will be used by the printer to determine how the data stream is processed. Some of the options are: print data exactly as received; print data per format; set up a format; and set number of lines per page. Characters may be double width (7 by 16). Formatting includes left, right, or decimal point alignment, as well as leading zero, floating dollar sign, and trailing minus sign.

Nothing much is happening as far as correction of system bugs with the exception of losing the cursor when hitting return. This can be corrected with a ROM replacement at a cost of $10.

Commodore Software still isn't being distributed yet, although an announcement was mailed in April indicating the first batch release was imminent.

Several users have written that their keytop surfaces are wearing off on the more heavily used keys. Does anyone know a preventive measure for this? Also, many questions about adding a previously saved routine on cassette to a program in memory. Anyone have any ideas on this?

In general, comments about the PET have been very favorable. In fact, the response range seems to be favorable to ecstatic, with no one yet indicating they were dissatisfied.

Software

James Johnson, 9304 Emory Grv. Road, Gaithersburg, MD 20760 has a Blackjack program which he says permits 5 players, double down, split, and uses full PET graphics. He will sell this program on cassette for $5.

C.M. Stuart & Co., 5115 Menefee Dr. Dallas, TX 75227 has developed a Casino Simulation Package containing Blackjack, Baccarat, Craps, and Roulette. All options available at the Casino of the MGM Grand Hotel in Las Vegas have been incorporated in the programs, which are available on cassette including complete rules at $9.95 each or $25 for all four.

Penninsula School Computer Project, Penninsula Way, Menlo Park, CA 94025 is selling 3 program tapes: (1) PILOT (CAI lan-

****

PET User Notes published 6 (or more) times a year by Gene Beals, PET User Group, Box 371, Montgomeryville, PA 18936. Membership/subscription on annual basis is $5 (U.S.) in U.S. and Canada, and $10 in other countries.

guage) and 5 sample programs $19.95 (2) Renumber, Lemon, Kaleidoscope, WSFN $14.95 (3) QUEST, DRAW (very nice expanded version) $9.95. Write them with SASE for order form.

Andy Fraley, 1753 York Rd., Reading, PA 19610 has four action games with graphics for sale at $5.95 each. They are Seawolf, Dogfight, Bomber, and Indy 500.

Several people suggested that we publish a PET software directory. If there is sufficient interest, we will attempt it. Mr. Elwood Anderson of Channel Data Systems, Goleta, CA suggested a format to include: cassette name; program names; publisher; author; price; memory req.; manual included; sold through (author, publisher, dealers, etc.); and brief program abstract. If you think this is a good idea, send your program info. We will publish as often as necessary to keep the listings current.

Hardware

International Technical Systems, Inc., Box 264, Woodbridge, VA 22194 announced a PEM-8K memory expansion module at $279. The 8K unit is assembled with case, power supply, and connecting hardware. ITS will include a 10K Mathematical/Financial software package with initial orders for PEM-8K.

Otherware

MJ Salisbury has a carrying case for PET which includes permanent foam cushioning material, removable hinged cover, and locking latches. $49.50 + shipping (COD). CASE, Box 6633, Oxnard, CA 93030.

Microsette, 777 Palomar Ave., Sunnyvale, CA 94086 has blank C-10 tapes in plastic box. Send $1.00 for sample tape.

TIS, Box 921, Los Alamos, NM 87544 has a series of workbooks on using the PET computer. Write them for further information.

Francis T. Chambers, Rock House, Ballycroy, Westport, Co. Mayo, Ireland has a winning strategy for Othello (Reversi). Many of you have seen the Othello program from the BYTE article, and know it needs help to play well. Frank has written a complete description of his method, and will air mail a copy for $5.

Software Exchange

Carl Bach, 1015 Saturn Springs Dr., Garland, TX 75041 has many game programs on cassette to exchange.

Microcomputer Resource Center, 5150 Anton Drive Room 212, Madison, WI 53719 is running a PET software exchange. Write them for submission format.

Errors in previous issue (V1, Issue 3)

page 1 -- line 12 in first paragraph should read 98 rather than 248.
page 9 -- the PET Character Code table was furnished by Frank Campbell. The original sent by Frank did not reproduce well, so we retyped it at the last minute. In so doing, we switched two keyboard symbols in the third column. The 103/39 and the 108/44 symbols should be interchanged.

Also, apologies to Jim Butterfield -- his Bagels program on page 6 has no indication of the continuation on page 11.

Programs on Tape

SEQUENCE -- you have to sort a character list
STATECAPS -- quiz on states and capitals from 3/78 Kilobaud
RACETRACK -- see listing in V.1, Issue 3 - RE McGee
DEFLECTION -- F. Dunlap
ADDITION -- math drill -- F. Dunlap, C. Pitcairn
OTHELLO -- from 9/77 BYTE
OTHELLO for 2 -- F. Dunlap
BAGELS -- like Mastermind - J. Butterfield
Cash Flow/Return on Investment - R. Goldsmith
STAR TREK
BLACKJACK -- D. Liem - nice graphics, humorous patter by dealer
LIFE -- 40*25 F. Covitz
LIFE -- 64*64 F. Covitz
FILE LIST & SORT -- from Kilobaud. Modified by Robert Harvey
TRAP -- trap the PET, avoid being trapped
Super Mastermind -- JR Marcou
Lunar Lander -- JR Marcou
NIM --J Butterfield
QUBIC -- 3D TIC-TAC-TOE
LIST MEMORY -- Charles Combs
Machine Language Monitor -- a good version - similar to the one Commodore will be sending
TIME -- large clock display
BIORYTHM -- Kenneth Finn - very concise code - plots with only 30 BASIC statements
KING -- business-social simulation or game
BREAKOUT -- simple version of the arcade game
SWATPLOT -- F. Campbell (see V.1 Issue 3) - plots in 80x50 format

$2 for the first program including cassette and postage, $1 for each additional -- max 4 per tape. Most of the programs will be listed in this newsletter, so the programs on tape service is available only to save you key entry time. It is priced to hopefully cover the cost of a part time person to copy and mail the tapes.

If you have a program to add to the list, please send it on tape. We will copy it and return your tape with several other programs.

****

Sensing Keys

Several people wrote that location 515 contained a key identifier code indicating whether a key is depressed during each monitor scan cycle. The value is 1 to 80 depending on the key, or 255 if no key is depressed.

```
10 K=PEEK (515): IF K=255 GOTO 10
```

Shift key depression can also be detected by checking location 515 (value is 1 if the key is depressed, 0 if it is not).

PET Matrix-Decoded Keyboard

Value of key = row value + column value

	16	8	15	7	14	6	13	5	12	4	11	10	2	9	1
64	!	"	#	$	%	'	&	\	(	)	←	ho	↑↓	⇆	del
48	Q	W	E	R	T	Y	U	I	O	P	↑	7	8	9	/
32	A	S	D	F	G	H	J	K	L	:	re	4	5	6	*
16	Z	X	C	V	B	N	M	,	;	?	re	1	2	3	+
0	sh	re	@	[	]	space		<	>	st	sh	0	.	-	=

Basic Command Abbreviations

C.W. Hennig of University of Waterloo found two letter entry codes for most of the Commodore BASIC commands. Generally, the entry abbreviations consist of the first letter of the command followed by the shifted second letter i.e. Go is GOTO, Pr is PRINT#1, Ve is VERIFY, Sp is SPC(, etc. If you are not in the lower case mode (POKE 59468,14) you will see the graphic character associated with the shifted entry of the letter rather than the lower case letter itself (I/ instead of In for INPUT#). In either case when you LIST, you will see the unabbreviated BASIC word, just as ? expands to PRINT.

Mr. Neil Harris carried Carl Hennig's discovery a step further by finding several 3 character entry abbreviations for some of the BASIC words not having 2 character entries.

Commodore BASIC Abbreviations

Command/ Statement	Abbreviation	Command/ Statement	Abbreviation
LET	Le	NOT	No
READ	Re	RUN	Ru
PRINT	?	CLR	Cl
DATA	Da	LIST	Li
THEN	Th	CONT	Co
FOR	Fo	FRE	Fr
NEXT	Ne	TAB(	Ta
DIM	Di	SPC(	Sp
END	En	PEEK	Pe
GOTO	Go	POKE	Po
RESTORE	REs	USR	Us
GET	Ge	SYS	Sy
GOSUB	GOs	WAIT	Wa
DEF	De	LEFT$	LEf
RETURN	REt	RIGHT$	Ri
STOP	St	MID$	Mi
STEP	STe	CHR$	Ch
SGN	Sg	ASC	As
ABS	Ab	VAL	Va
SQR	Sq	STR$	STr
RND	Rn	OPEN	Op
SIN	Si	CLOSE	CLo
COS	Co	SAVE	Sa
TAN	Ta	LOAD	Lo
ATN	At	VERIFY	Ve
LOG	Lo	PRINT#	Pr
EXP	Ex	INPUT#	In
AND	An	CMD	Cm

****

INPUT Without Data

In response to some of the questions on hitting RETURN without data being entered and having control return to the operating system, try one of these:

```
100 OPEN 1,0,1
200 INPUT#1,A$:PRINT
300 IF A$="" GOTO 200
```

If you hit return with no data, a null value is returned.

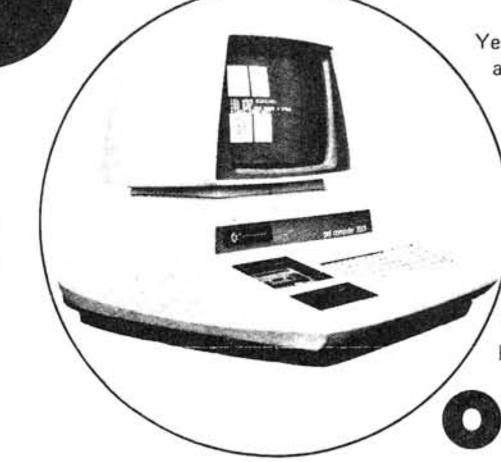
```
100 OPEN 1,0,1
200 INPUT#1 A:PRINT
```

In this example, return with no data enters 0. You can still return to the operating system if you enter non-numeric data (BAD DATA ERROR).

```
100 INPUT "ENTER VALUEbbb▚clclcl";A$
200 IF A$="   " THEN PRINT "cucu": GOTO 100
```

This method prints a character (in this case the shifted question mark) and then places the cursor on that character ready for input. If return without data, then you will go back for another try.

PET 2001



Yes! — It's hard to believe. The PET-2001, a full-fledged computer by Commodore. What do you get? Full ASCII keyboard, 9-inch CRT and a tape cassette all in one lightweight unit. Fully-programmable in extended BASIC (20% faster than most other BASICs).

14k ROM (Read Only Memory). 8K RAM (Random Access Memory). Expandable to 32K. PET's very own graphic instruction set.

For HOME/OFFICE/GAMES $795 (8K) Portable, Affordable, and Unbelievable.

ONLY $795

PET SOFTWARE NOW AVAILABLE!

MORTGAGE ($15.95) This program calculates mortgage information when provided with certain basic data. Calculated information includes: Principal paid to date; Interest paid to date; Total of Payments paid to date; Outstanding principal and interest; Total payments remaining; Mortgage equity; Interest and principal breakdown for any month. **CHECKBOOK ($15.95)** A cash receipts and disbursements program that will make it a breeze to keep accounts and up to date records. Checks can be searched and sorted by type, e.g., medical, legal, tax-deductible, rent, food, etc. **FINANCE (12.95)** A variety of useful financial formulas in one simple easy to use program. Includes: Compound interest; Discounts; Nominal and effective interest; Annuities; Loans; Depreciation;Earned interest and much more.
ANNUAL REPORT ANALYZER ($22.95) With Annual Report in hand, you input revenue and income figures for previous five years (estimated earnings, too, if you wish) as well as basic Balance Sheet data. This Street Ware program computes: Percentage year-to-year growth in sales, profits, and earnings per share; Average earnings per share and compound earnings per share over 5 years; PE Ratio; Profit margin for previous 5 years, with a graphic display that plots revenues against profit margins; Current ratio; Book value; Return on equity; Debt to equity ratio ; Payout ratio; Dividend yield; Implied growth rate; Implied total return; Theoretical PE ratio; Theoretical value for stock.
STOCK ANALYZER ($34.95) This tape includes a copy of ANNUAL REPORT ANALYZER on reverse side. The program is essentially the same except that data is automatically read from Data Base tapes simply by entering ticker symbols. **DATA BASE (Updated monthly; total of 12 tapes per year) ($175.00)** Includes statistical data on over 2.500 Industrial Stocks on the New York, American, and Over the Counter Exchanges. Data base tapes are updated monthly by stock exchange on a rotating basis, i.e., twelve tapes per year. Data includes: Ticker symbol, Corporate name, Industrial classification; Revenues, earnings, and earnings per share for previous 5 years; Estimated earnings for current year; Shares outstanding, current assets, current liabilities; Dividends, long-term debt. **OPTIONS ($24.95)** The National Corporate Sciences' version of the Black-Scholes equation, this program computes the theoretical value of an option. The program can be used to equal advantage by both options buyers and options writers. Value of option is graphically depicted by movement in stock price and days to expiration. **BONDS ($9.95)** A variety of bond programs to calculate interest and yield to maturity, present value and future value of bonds, effective yield, and basis price of bonds.

This fantastic low-cost Business System utilizes the power of the PET-2001 Model 8K Computer along with the Brand New PET 120 cps Impact Printer (makes 4-5 copies) and an additional Digitally Controlled Tape Drive —for about $1500. Accounts Receivable and Inventory Control software is available for the Pet Business System!

PET BUSINESS SYSTEM

PET PERIPHERALS NOW AVAILABLE!

PET to RS-232 interface— PET to telephone coupler— PET to S-100 Interface— PET to Large Video Screen— PET High Speed Printer— PET Tape Drive— PET Music Synthesizer and more. . .

THE COMPUTER FACTORY
790 MADISON AVENUE, NEW YORK, NY 10021
(212) 249-1666 or (212) PET 2001 T-F 10-6 Sat. 10-4

BANKAMERICARD

LEASING AVAILABLE

****

Day of Week Routine
Earl H. Wuchter Catasauqua, PA

This routine will return the day of week for any given day from Mar 1, 1900 through Feb 29, 2000, and can be expanded to work for any date. The routine does not destroy the date, and may be used as a subroutine after the WD$ array is initialized.

```
10 REM INITIALIZE ARRAY
12 DIM WD$(6)
14 DATA "SUN","MON","TUE","WED","THU","FRI","SAT"
16 FOR J=0 TO 6: READ WD$(J): NEXT
18 REM INPUT DATE
20 PRINT"ENTER MONTH, DAY, YEAR";: INPUT MN,DY,YR
22 REM WEEKDAY PGM BEGINS HERE. MAY BE A SUB
24 REM ADJUST YEAR TO BEGIN ON MAR 1
26 CY=YR:M=MN-2
28 IF M<1 THEN M=M+12: CY+CY-1
30 Y=CY-INT(CY/100)*100: REM Y=LAST TWO DIGITS
32 REM C=INT(CY/100) WHEN USING EXPANDED PGM
34 REM WD=(Y+Y/4+C/4-2*C+DY+(2.6*M-.2))MOD7
36 WD=Y+INT(Y/4)+1+DY+INT(2.6*M-.1999)
38 WD=WD-INT(WD/7)*7
40 PRINT WD$(WD);
```

Program notes:
REM line 34 shows (2.6*M-.2)
Code line 36 uses (2.6*M-.1999)
.2 is correct, but something less must be used because of the internal floating point roundoff.
REM line shows +C/4-2*C
Code line uses +1
For the years covered by this version, C is always equal to 19.
INT(19/4)-2*19=-34 -34 MOD7=+1
This simplifies the MOD function on line 38 by getting rid of the negative number. To expand the pgm for any year, break the year into C and Y after it is adjusted to begin on Mar 1. You will need a routine then (to replace line 38) that will handle negative numbers.

LIFE for the PET
Frank Covitz Lebanon, New Jersey

Since this is the first time I have attempted to set down a machine language program for the public eye, I will attempt to be as complete as practical without overdoing it.

The programs I will document here are concerned with the game of LIFE, and are written in 6502 machine language specifically for the PET 2001 (8K version). The principles apply to any 6502 system with graphic display capability, and can be debugged (as I did) on non-graphic systems such as the KIM-1.

The first I heard of LIFE was in Martin Gardner's "Recreational Mathematics" section in Scientific American, Oct, Nov 1970, Feb 1971. As I understand it, the game was invented by John H. Conway, an English mathematician. In brief, LIFE is a "cellular automation" scheme, where the arena is a rectangular grid (ideally of infinite size). Each square in the grid is either occupied or unoccupied with "seeds", the fate of which are governed by relatively simple rules, i.e. the "facts of LIFE". The rules are: 1. A seed survives to the next generation if and only if it has two or three neighbors (right, left, up, down, and the four diagonally adjacent cells) otherwise it dies of loneliness or overcrowding, as the case may be. 2. A seed is born in a vacant cell on the next generation if it has exactly 3 neighbors.

With these simple rules, a surprisingly rich game results. The original Scientific American article, and several subsequent articles reveal many curious and surprising initial patterns and results. I understand that there even has been formed a LIFE group, complete with newsletter, although I have not personally seen it.

The game can of course be played manually on a piece of graph paper, but it is slow and prone to mistakes, which have usually disastrous effects on the final results. It would seem to be the ideal thing to put to a microprocessor with bare-bones graphics, since the rules are so simple and there are essentially no arithmetic operations involved, except for keeping track of addresses and locating neighbors.

As you know, the PET-2001 has an excellent BASIC interpreter, but as yet very little documentation on machine language operation. My first stab was to write a BASIC program, using the entire PET display as the arena (more about boundaries later), and the filled circle graphic display character as the seed. This worked just fine, except for one thing - it took about 2 1/2 minutes for the interpreter to go through one generation! I suppose I shouldn't have been surprised since the program has to check 8 neighboring cells to determine the fate of a particular cell, and do this 1000 times to complete the entire generation (40x25 characters for the PET display).

The program below is a 6502 version of LIFE written for the PET. It needs to be POKE'd into the PET memory, since I have yet to see or discover a machine language monitor for the PET. I did it with a simple BASIC program and many DATA statements (taking up much more of the program memory space than the actual machine language program!). A routine for assembling, and saving on tape machine language programs on the PET is sorely needed.

The program is accessed by the SYS command, and takes advantage of the display monitor (cursor control) for inserting seeds, and clearing the arena. Without a serious attempt at maximizing for speed, the program takes about 1/2 second to go through an entire generation, about 300 times faster than the BASIC equivalent! Enough said about the efficiency of machine language programming versus BASIC interpreters? BASIC is great for number crunching, where you can quickly compose your program and have plenty of time to await the results.

Program - LIFE 40*25

Address	Op.code	Byte 1,2	Addr.	Op.	B1	B2	Comments
MAIN	PHP PHA	TXA	1900	08	48	8A	Save A,X,Y,S
	PHA TYA	PHA	1903	48	98	48	
	TSX TXA	PHA	1906	BA	8A	48	
	CLD		1909	D8			Clear dec. mode
	JSR	INIT	190A	20	30	19	
	JSR	SCR-TEMP	190D	20	8A	19	
GEN	JSR	TEMP-RCS	1910	20	E6	19	
	JSR	GENER	1913	20	00	1A	
	JSR	TEMP-SCR	1916	20	70	19	
	INCz	Times	1919	E6	38		Repeat 255 times
	BNE	GEN	191B	D0	F3		before quitting
	PLA TAX	TXS	191D	68	AA	9A	Restore S,Y,X,A
	PLA TAY	PLA	1920	68	A8	68	
	TAX PLA	PLP	1923	AA	68	28	
	JMP	BASICON	1926	4C	8B	C3	Return to BASIC

```
Address   Op.code   Byte 1,2          Addr.  Op.  B1 B2   Comments

INIT      LDXim     19                1930   A2   19      Read in 25dec init.
LOAD      LDAx      Data-1            1932   BD   3A 19   values
          STAzx     1Fhex             1935   95   1F      Store them in p. zero
          DEX                         1937   CA
          BNE       LOAD              1938   D0   F8
          RTS                         193A   60
=====================================================================

DATA      ScrL  ScrH  ChL             193B   00   80 00
          ChH   ScrLo ScrHo           193E   15   00 80
          TempL TempH TempLo          1941   00   1B 00
          TempHo Up   Down            1944   1B   D7 28
          Right Left  Ur              1947   01   FE D8
          Ul    Lr    Ll              194A   D6   29 27
          N  ScrlastL ScrlastH        194D   00   E8 83
          RcsLo RcsHo Tmp             1950   00   15 00
          Times                       1953   00
=====================================================================

TEMP→SCR  JSR       RESTORE           1970   20   A6 19   Get init. addresses
LOAD      LDAiy     TempL             1973   B1   26      Fetch byte from Temp
          BNE       ONE               1975   D0   06      Branch if ≠ 0
          LDAim     32dec             1977   A9   20      The symbol for blank
          STAiy     ScrL              1979   91   20      Dump it to screen
          BNE       NEXT              197B   D0   04
ONE       LDAim     81dec             197D   A9   51      The "dot" symbol
          STAiy     ScrL              179F   91   20      Dump it to screen
NEXT      JSR       NEXTADDR          1981   20   BD 19   Fetch next addr.
          BEQ       LOAD              1984   F0   ED
          JSR       RESTORE           1986   20   A6 19   Restore init addr's.
          RTS                         1989   60
=====================================================================

SCR→TEMP  JSR       RESTORE           198A   20   A6 19   Get init. addresses
LOAD      LDAiy     ScrL              198D   B1   20      Read data fr. screen
          CMPim     81dec             198F   C9   51      Is it a "dot"?
          BEQ       ONE               1991   F0   06      Branch if "dot"
          LDAim     00                1993   A9   00      Otherwise it's blank
          STAiy     TempL             1995   91   26      Store it
          BEQ       NEXT              1997   F0   04      Uncond. branch
ONE       LDAim     01                1999   A9   01      A "dot" was found
          STAiy     TempL             199B   91   26      Store it
NEXT      JSR       NEXTADDR          199D   20   BD 19   Fetch next addr.
          BEQ       LOAD              19A0   F0   EB
          JSR       RESTORE           19A2   20   A6 19   Restore init addr's
          RTS                         19A5   60
=====================================================================

RESTORE   LDAim     00                19A6   A9   00      Zero A,X,Y
          TAX  TAY                    19A8   AA   A8
          STAz      ScrL              19AA   85   20      Initialize ScrL,
          STAz      TempL             19AC   85   26      TempL,RcsL,ScrH,
          STAz      RcsL              19AE   85   39      TempH,RcsH
          LDAz      ScrHo             19B0   A5   25
          STAz      ScrH              19B2   85   21
          LDAz      TempHo            19B4   A5   29
          STAz      TempH             19B6   85   27
          LDAz      RcsHo             19B8   A5   36
          STAz      RcsH              19BA   85   3A
          RTS                         19BC   60
=====================================================================

NEXTADDR  INCz      TempL             19BD   E6   26      Get next low order
          INCz      ScrL              19BF   E6   20      byte address
          INCz      RcsL              19C1   E6   39
          INX                         19C3   E8
          CPXz      ScrlastL          19C4   E4   33      Is it the last?
          BEQ       PAGECH            19C6   F0   0C      Is it the last page?
          CPXim     00                19C8   E0   00      Is it a page boundary?
          BNE       LOAD              19CA   D0   0E      If no , then not done
          INCz      TempH             19CC   E6   27      Otherwise advance to
          INCz      ScrH              19CE   E6   21      next page
          INCz      RcsH              19D0   E6   3A
          BNE       LOAD              19D2   D0   06      Uncond. branch
PAGECH    LDAz      ScrlastH          19D4   A5   34      Check for last
          CMPz      ScrH              19D6   C5   21      page
          BEQ       DONE              19D8   F0   03      If yes, then done
LOAD      LDAim     00                19DA   A9   00      Return with A=0
          RTS                         19DC   60
DONE      LDAim     01                19DD   A9   01      Return with A=1
          RTS                         19DF
=====================================================================

TEMP→RCS  JSR       RESTORE           19E6   20   A6 19   Initialize addr's
LOAD      LDAiy     TempL             19E9   B1   26      Fetch data from Temp
          BNE       ONE               19EB   D0   06      If ≠ 0, then it's alive
          LDAim     32dec             19ED   A9   20      The symbol for blank
          STAiy     RcsL              19EF   91   39      Store it in screen copy
          BNE       NEXTADDR          19F1   D0   04      Then on to next addr.
ONE       LDAim     81dec             19F3   A9   51      The "dot" symbol
          STAiy     RcsL              19F5   91   39      Store it in screen copy
```

```
Address   Op.code    Byte 1,2     Addr.  Op. B1 B2   Comments

NEXTADDR  JSR        NEXTADDR     19F7   20  BD 19   Fetch next addr.
          BEQ        LOAD         19FA   F0  ED      If A=0, then not done
          JSR        RESTORE      19FC   20  A6 19   Restore init. addr's
          RTS                     19FF   60
=====================================================================

GEN       JSR        RESTORE      1A00   20  A6 19   Initialize addr's
AGAIN     JSR        NBRS         1A03   20  2F 1A   Fetch # of neighbors
          LDAiy      RcsL         1A06   B1  39      Fetch current data
          CMPim      81dec        1A08   C9  51      Is it a "dot"?
          BEQ        OCC          1A0A   F0  0C      If yes, then branch
          LDAz       N            1A0C   A5  32      Otherwise it's blank
          CMPim      03           1A0E   C9  03      and we need to check
          BNE        NEXTADDR     1A10   D0  14      for a birth
BIRTH     LDAim      01           1A12   A9  01      It gives birth!
          STAiy      TempL        1A14   91  26      Store it in Temp
          BNE        NEXTADDR     1A16   D0  0E      Uncond. branch
OCC       LDAz       N            1A18   A5  32      Fetch # of neighbors
          CMPim      03           1A1A   C9  03      If it has 3 or 2
          BEQ        NEXTADDR     1A1C   F0  08      neighbors, it survives
          CMPim      02           1A1E   C9  02
          BEQ        NEXTADDR     1A20   F0  04
DEATH     LDAim      00           1A22   A9  00      It died!
          STAiy      TempL        1A24   91  26      Store it in Temp
NEXTADDR  JSR        NEXTADDR     1A26   20  BD 19   Fetch next address
          BEQ        AGAIN        1A29   F0  D8      If = 0, then not done
          JSR        RESTORE      1A2B   20  A6 19   Restore init. addr's
          RTS                     1A2E   60
=====================================================================

NBRS      TYA  PHA  TXA           1A2F   98  48 8A   Save Y and X on stack
          PHA                     1A32   48
          LDYim      00           1A33   A0  00      Set Y and N = 0
          STYz       N            1A35   84  32
          LDXim      08           1A37   A2  08      Check 8 neighbors
OFFS      LDAzx      Offset-1     1A39   B5  29      Fetch offset data
          BPL        ADD          1A3B   10  15      Add if offset is pos.
          EORim      FF           1A3D   49  FF      Otherwise get set to
          STAz       Tmp          1A3F   85  37      subtract
          SEC                     1A41   38
          LDAz       RcsL         1A42   A5  39
          SBCz       Tmp          1A44   E5  37      Subtract to get the
          STAz       ChL          1A46   85  22      correct neighbor
          LDAz       RcsH         1A48   A5  3A      address
          STAz       ChH          1A4A   85  23
          BCS        EXAM         1A4C   B0  11      OK, find out what's
          DECz       ChH          1A4E   C6  23      Page cross. there
          BNE        EXAM         1A50   D0  0D      Unc. branch
ADD       CLC                     1A52   18          Get set to add
          ADCz       RcsL         1A53   65  39      Add
          STAz       ChL          1A55   85  22      Store the low part
          LDAz       RcsH         1A57   A5  3A      Fetch the high part
          STAz       ChH          1A59   85  23      Store it
          BCC        EXAM         1A5B   90  02      OK, what's there
          INCz       ChH          1A5D   E6  23      Page crossing
EXAM      LDAiy      ChL          1A5F   B1  22      Fetch the neighbor
          CMPim      81dec        1A61   C9  51      data byte, and see
          BNE        NEXT         1A63   D0  02      if it's occupied
          INCz       N            1A65   E6  32      Accumulate # of nbrs.
NEXT      DEX                     1A67   CA
          BNE        OFFS         1A68               Not done
          PLA  TAX  PLA           1A6A   68  AA 68   Restore X,Y from stack
          TAY  RTS                1A6D   A8  60
```

The program may be broken down into manageable chunks by subroutining. There follows a brief description of the salient features of each section:

MAIN (hex 1900)

In a fit of overcaution (since this was the first time I attempted to write a P.E.T. machine language program) you will notice the series of pushes at the beginning and pulls at the end. I decided to save all the internal registers in the stack in page 1, and also included the CLD (clear decimal mode) just in case. Then follows a series of subroutine calls to do the LIFE generation and display transfers. Notice that the all-capital symbols refer to specifically labelled addresses, and the first letter capitalized symbols refer to zero page locations. The zero page location, Times, is a counter to permit several loops through LIFE before returning. As set up, Times is initialized to zero (hex location 1953), so that it will loop 256 times before jumping back. This of course can be changed either initially or while in BASIC via the POKE command. The return via JMP BASICON (4C 8B C3) may not be strickly orthodox, but it seems to work OK.

INIT (hex 1930) and DATA (hex 193B)

This shorty reads in the constants needed, and stores them in page zero. Scr refers to the PET screen, Temp is a temporary working area to hold the new generation as it is evolved, and Rcs is essentially a copy of the PET screen data, which I found to be necessary to avoid "snow" on the screen during the read/write operations directly on the screen locations. Up, Down, etc. are the offsets to be added or subtracted from an address to get all the neighbor addresses. The observant reader will note the gap in the addresses between some of the routines.

TEMP→SCR (hex 1970)

This subroutine quickly transfers the contents of Temp and dumps it to the screen, using a dot (81_{dec}) symbol for a live cell (a 1 in Temp) and a space (32_{dec}) for the absence of a live cell (a 0 in Temp).

SCR→TEMP (hex 198A)

This is the inverse of TEMP→SCR, quickly transfering (and encoding) data from the screen into Temp.

RESTORE (hex 19A6)

This subroutine fetches the initial addresses (high and low) for the Scr, Temp, and Rcs memory spaces.

NEXTADDR (hex 19BD)

Since we are dealing with 1000 bytes of data, we need a routine to increment to the next location, check for page crossing (adding 1 to the high address when it occurs), and checking for the end. The end is signaled by returning a 01 in the accumulator, otherwise a 00 is returned via the accumulator.

TEMP→RCS (hex 19E6)

The Rcs address space is a copy of the screen, used as mentioned before to avoid constant "snow" on the screen if the screen were being continually accessed. This subroutine dumps data from Temp, where the new generation has been computed, to Rcs.

GEN (hex 1A00)

We finally arrive at a subroutine where LIFE is actually generated. After finding out the number of neighbors of the current Rcs data byte from NBRS, GEN checks for births (CMPim 03 at hex addr. 1A0E) if the cell was previously unoccupied. If a birth does not occur, there is an immediate branch to NEXTADDR (the data byte remains 00). If the cell was occupied (CMPim 81_{dec} at hex 1A08), GEN checks for survival (CMPim 03 at hex 1A1A andCMPim 02 at hex 1A1E), branching to NEXTADDR when these two conditions are met, otherwise the cell dies (LDAim 00 at hex 1A22). The results are stored in Temp for the 1000 cells.

NBRS (hex 1A2F)

NBRS is the subroutine that really does most of the work and where most of the speed could be gained by more efficient programming. Its job, to find the total number of occupied neighbors of a given Rcs data location, is complicated by page crossing and edge boundaries. In the present version, page crossing is taken care of, but edge boundaries (left, right, top, and bottom of the screen) are somewhat "strange". Above the top line and below the bottom line are considered as sort of forbidden regions where there should practically always be no "life" (data in those regions are not defined by the program), but I have found that there has never been a case where 81's have been present (all other data is considered as "unoccupied characters). The right and left edges are different, however, and lead to a special type of "geometry". A cell at either edge is not considered as special by NBRS, and so to the right of the right-edge location is the next sequential address. On the screen this is really the left edge location, and one line lower. The inverse is true, of course for left addresses of left-edge locations. Topologically, this is equivalent to a "helix". No special effects of this are seen during a simple LIFE evolution, since it just gives the impression of disappearing off one edge while appearing on the other edge. For an object like the "spaceship" (see Sci. Am articles), then, the path eventually would cover the whole LIFE arena. The fun comes in when a configuration spreads out so much that it spills over both edges, and interacts with itself. This, of course cannot happen in the infinite universe, so that some of the more complex patterns will not have the same fate in the present version of LIFE. Most of the "blinkers", including the "glider gun" come out OK.

This 40x25 version of LIFE can undoubtedly be made more efficient, and other edge algorithms could be found, but I chose to leave it in its original form as a benchmark for my first successfully executed program in writing machine language on the PET. One confession, however - I used the KIM-1 to debug most of the subroutines. Almost all of them did not run on the first shot! Without a good understanding of PET memory allocation particularly in page zero I was bound to crash many times over, with no recovery other than pulling the plug. The actual BASIC program consisted of a POKING loop with many DATA statements (always save on tape before running!).

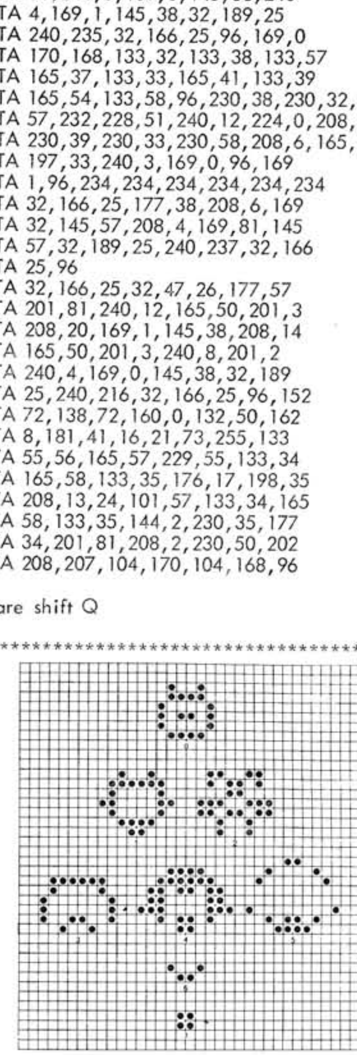
[ed. note: Frank Covitz also sent a 64x64 version of LIFE which we will be printing next issue. Frank had to wade through deciphering PET structure with no machine language monitor. He said he might do this version of LIFE differently if he had to do it over, but regardless, many thanks to him. I think the resulting program is extremely good.]

LIFE by Frank Covitz

```
5     A0=6459
10    FOR I=0 TO 307: READ A%:POKE A0+I,A%: NEXT I
20    RESTORE
30    PRINT "CLEAR SCREEN, THEN ENTER LIFE SEEDS (●),
THEN 'HOME','GOTO 100'":END
100   POKE 1,59:POKE 2,25
110   X=USR(0)
120   GET C$
130   IF C$="S" THEN END
140   POKE 1,66
150   GOTO 110
6459  DATA 216,32,76,25,32,138,25
6466  DATA 32,230,25,32,0,26,32,112,25,96
6476  DATA 162,25,189,86,25,149,31,202
6484  DATA 208,248,96,0,128,0,21,0
6492  DATA 128,0,27,0,27,215,40,1
6500  DATA 254,216,214,41,39,0,232,131
6508  DATA 0,21,0,0
6512  DATA  32,166,25,177,38,208,6,169
6520  DATA 32,145,32,208,4,169,81,145
6528  DATA 32,32,189,25,240,237,32,166
6536  DATA 25,96,32,166,25,177,32,201
6544  DATA 81,240,6,169,0,145,38,240
6552  DATA 4,169,1,145,38,32,189,25
6560  DATA 240,235,32,166,25,96,169,0
6568  DATA 170,168,133,32,133,38,133,57
6576  DATA 165,37,133,33,165,41,133,39
6584  DATA 165,54,133,58,96,230,38,230,32,230
6592  DATA 57,232,228,51,240,12,224,0,208,14
6600  DATA 230,39,230,33,230,58,208,6,165,52
6608  DATA 197,33,240,3,169,0,96,169
6616  DATA 1,96,234,234,234,234,234,234
6624  DATA 32,166,25,177,38,208,6,169
6632  DATA 32,145,57,208,4,169,81,145
6640  DATA 57,32,189,25,240,237,32,166
6648  DATA 25,96
6656  DATA 32,166,25,32,47,26,177,57
6664  DATA 201,81,240,12,165,50,201,3
6672  DATA 208,20,169,1,145,38,208,14
6680  DATA 165,50,201,3,240,8,201,2
6688  DATA 240,4,169,0,145,38,32,189
6696  DATA 25,240,216,32,166,25,96,152
6704  DATA 72,138,72,160,0,132,50,162
6712  DATA 8,181,41,16,21,73,255,133
6720  DATA 55,56,165,57,229,55,133,34
6728  DATA 165,58,133,35,176,17,198,35
6736  DATA 208,13,24,101,57,133,34,165
6744  DATA 58,133,35,144,2,230,35,177
6752  DATA 34,201,81,208,2,230,50,202
6760  DATA 208,207,104,170,104,168,96
```

life seeds are shift Q

****



The Cheshire cat (0) fades to a grin (6) and disappears, leaving a paw print (7)

PET Machine Language Programming
Raynor Taylor Charleston, SC

Try this little program

```
10 FOR A=6400 TO 6409
20 READ B
30 POKE A,B
40 NEXT A
50 SYS(6400)
60 DATA 162,0,138,157,80,129,232,208,249,96
70 END
```

Intrigued? Read on!

By just looking at the program in BASIC, you can see that the ten data elements are read and poked (written) into memory starting at decimal location 6400. These data elements make up a machine language subroutine that is executed by the statement in line 50. As you have probably already found out by running the BASIC program, the machine language subroutine causes all the printable characters to appear about half way down on the screen.

Below is a translation of the ten data elements into an easily read English-type format.

Step	Address in Memory Base 16	Instruction Codes Base 16		Mnemonic	Addressing Mode
#1	$1900	A2 00		LDX	#$00
#2	$1902	8A	LP	TXA	
#3	$1903	9D 50 81		STA	$8150, X
#4	$1906	E8		INX	
#5	$1907	D0 F9		BNE	LP
#6	$1909	60		RTS	

The 6502 microprocessor in the PET, located at coordinates F-3 on the main board inside, basically has three registers -- A, X, Y. This program makes use of two, the A and X. Going through the program step by step, this is what happens:

Step	Effect
#1	Load zero into the X register
#2	Transfer X to A
#3	Store A at location $8150+X (loc. $8150 is the eight line on the screen
#4	Add one to the X register
#5	Jump to step #2 if X does NOT equal zero (the X register starts all over at zero once the value of 256 is reached - thus this loop is repeated 256 times)
#6	Return to BASIC program

This machine language program uses only six of the 6502's 148 instructions. The six step program is also very small compared to the approximately 7000 step program in the PET's permanent memory. Nevertheless, the only way to learn about the PET's true brain is to write and experiment with small programs. A very helpful tool in writing machine language programs is the monitor. The monitor accepts your input, changes it into a computer readable form, and changes the computer's jibberish into the clear format illustrated above. You may find that some of your programs bomb -- that is just don't work and cause the computer to crash (go crazy) -- but don't worry, it happens to the best of us.

[ed. note: Raynor Taylor's program notes provide a good starting point if you haven't been exposed to machine language programming. His machine language program uses the X register as a counter, as an offset to determine the address in screen memory where the character will be stored, and as the character value itself. In step #5 the branch uses a relative address of -7 (1909-1902), or F9 in single byte 2 complement mode.

In the BASIC program, lines 10 through 30 store the DATA values in hexadecimal locations 1900 to 1909 (6400 to 6409 decimal). The program then branches to 6400 with the SYS(6400). The machine language routine executes and returns to the calling program. An approximate BASIC equivalent subroutine would be:

```
1000 I=0
1100 POKE 33104+I,I
1200 I=I+1
1300 IF I<256 GOTO 1100
1400 RETURN
```

In Raynor's DATA statement line 60, he first had to convert the values in locations 1900 through 1909 to their decimal equivalent to obtain the data elements. To make this task easier, he has written a machine language monitor which he is selling for $7.50 along with some sample programs (biorythm, pi calculation, a target game, and two others) and complete documentation. Write him at Qtrs 718, Charleston Naval Base, Charleston, SC 29408.]

****

Notes from Chuck Stuart Dallas, TX

PET Random Number Generator

Some mention was made in the first issue of PET User Notes about problems associated with the Random Number Generator (RNG). One problem that I have noticed that was not mentioned is the fact that, on power-up, the RNG always returns the same sequence of Random Numbers when any positive integer seed is used.

This becomes a problem when using the RNG in games and simulations because the sequence is predictable. For example, you come home each day after work, you turn your PET on and load in your favorite game, say Blackjack. It doesn't take long to realize that the first two hands dealt are always the same. And if these hands are played the same each day, then the second hands, etc., will also be the same.

Fortunately, by using the PET's TI$ function, this predictability can be totally eliminated. The following line of code should be inserted just prior to the line that generates the random number for your program:

```
10 FOR I=1 TO VAL(RIGHT$(TI$,2)):R=RND(1):NEXT I
```

The odds against getting the same first number from power-up twice in a row are now 60 to 1. And the odds against getting the same first four cards in your favorite Blackjack game two days in a row are now 28,561 to 1. Your PET RNG will now generate truly random numbers, even just after power-up.

Floating Dollar Sign

Because of the leading space in front of any numeric character, some string manipulation is required in order to float a "$" or anything else just to the left of a number in the PET. The following routine can be used for this purpose.

```
10  N=100
20  PRINT "I WON"; : GOSUB 100: PRINT " AT THE RACES."
30  END
100 X=1: IF N<0 THEN X=0
110 PRINT" $"+RIGHT$(STR$(N),LEN(STR$(N))-X);
120 RETURN

RUN
I WON $100 AT THE RACES.
```

Lines 100 through 120 float the dollar sign. 'X' is required because when a number is negative, a minus sign replaces the leading space.

****

Using the IEEE Port as a General Purpose Port
JR Kinnard Austin, TX

I think that many PET owners will find as I have that a single 8 bit user port is not satisfactory for many applications and that the IEEE-488 interface is too complex to use with home built hardware. Fortunately the IEEE port can be used very much like the User Port and has several advanatages.

Data input to the IEEE DIO lines must be at TTL levels and is buffered into the 'A' side of an MCS6520 Peripheral Interface Adapter (PIA). Input data can be read from the DIO lines by a PEEK(59424). Data to be output on the DIO lines is buffered out of the 'B' side of the PIA. A POKE 59426,Data will transfer the data to the port. The DIO lines have a current sinking capability of 48 MA - much more than the 1.6 MA of the User Port. You must write all ones (255) into the output buffer before an input operation or a combination of input and output data will be read.

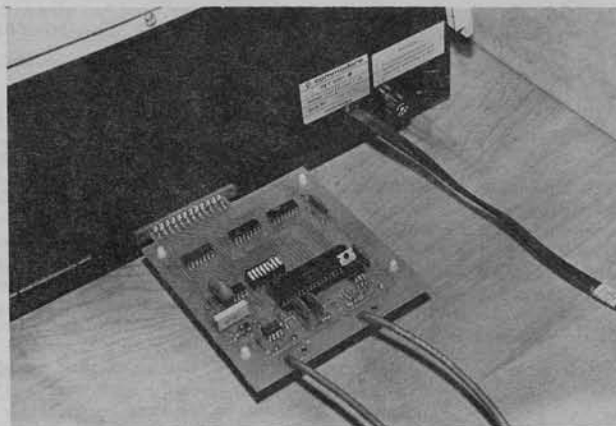
Output lines CA2 and CB2 are available from the PIA and are labeled NDAC and DAV, respectively, at the IEEE port. CA2 can be set to a low level by a POKE 59425,52 or to a high level by POKE 59425,60. CB2 can be controlled in the same way at address 59427.

Two other output lines on the IEEE port are PB1 and PB3 from the MCS6522 that drives the User Port. PB1 is labeled NRFD and PB3 is ATN on the IEEE connector. Outputs to these lines are bits 1 and 3 (values 2 and 8) of the register at address 59456.

CONNECTICUT microCOMPUTER

150 POCONO ROAD - BROOKFIELD, CONNECTICUT 06804

(203) 775-9659



RS-232 PRINTER ADAPTER FOR THE COMMODORE PET

Connecticut microComputer announces the first in a line of peripheral adapters for the COMMODORE PET. The PET ADApter model 1200 drives an RS-232 printer from the PET IEEE-488 bus. The PET ADA 1200 allows the PET owner to obtain hard copy program listings, and to type letters, manuscripts, mailing labels, tables of data, etc., using a standard RS-232 printer.

The PET ADA model 1200 is available assembled and tested, without power supplies, case, or RS-232 connector for $98.50 (plus $5.00 for shipping and handling) or complete for $169 (plus $5.00 for shipping and handling). Specify baud rate when ordering. (300 baud is supplied unless otherwise requested. Instructions for changing baud rate are included.)

This flyer was composed on the COMMODORE PET and printed on a GE Terminet 300 using PET ADA model 1200.

Send to:Connecticut microComputer;150 Pocono Rd;Brookfield, CT 06804

Qnty	Description	baud rate	price	total
	PET ADA 1200 (basic)		$98.50	
	PET ADA 1200 (complete)		$169.00	
			Subtotal	
			Connecticut residents add 7% sales tax	
			Handling and shipping - add per order	$5.00
			Foreign air mail - add $5.00 per order	
			Total enclosed	$

Name ______________________________

Address ______________________________

____

City, State ______________________ ZIP __________

Charge my BAC/VISA __________ MASTERCHARGE __________ Interbank No. __________

Card No. ______________________________ Exp. Date __________

Everything you always wanted to plug into your PET or APPLE*

*(but were afraid you couldn't find)

We're the new guys on the block and we've got a plan: to build the most extensive line of home computer peripherals and software available in this country. We're scouring every corner to find the best and the latest. And we plan to bring 'em to you first. Right now, we've got a healthy start with Pet and Apple and we're working on the TRS-80. If you don't see something here that you want, call us anyway. This is one mailing list you'll want to be on. Promise.

HARDWARE

Anderson-Jacobsen 841 I/O terminal*
- EBCD Keyboard or Correspondence $995.00

Pennywhistle 103 Modem
- Kit $129.95
- Assembled $179.95

Centronics Microprinter
- P1 (Parallel Interface) . . (Special) $495.00
- S1 (R5232C Serial Interface . . . (Special) $595.00

Parallel Interface Board . $180.00

Serial Interface Board . . $70.00

Pet or Apple Joystick with sample games tape
- Kit $29.95
- Assembled $39.95

SOFTWARE

PET
- House Break your Pet . . $19.95
- Star Trek $29.95
- Star Pilot $19.95
- Biorythm $19.95
- The Dragon $9.95
- Black Jack $19.95
- Artillery $14.95
- LEM $14.95
- Hangman $14.95
- Five Card Draw $14.95
- Deflection $10.00
- Home Data Retrieval . . $10.00
- Stat $20.00

APPLE
- U-Draw, 16K $17.95
- Tank War, 12K $12.95
- Number Cruncher $9.95
- Escape.. .... $12.95
- Dial-A-Maze $14.95
- Air Fight......... $12.95

8-Way Joystick Package (with Maze and Breakout Games $39.95
Extra-Joystick (with two player games) $19.95
(Also Available for Apple and TRS-80)

Shipping/handling: **$3** each unit hardware, $1 each unit software. No COD's or PO Boxes. Prices are cash only; credit card prices differ. Subject to availability & guaranteed by manufacturer. *FOB San Jose, CA.

Toll free Microline for credit card orders:

800-523-4550

In Pennsylvania and Canada
(215) 665-1112

Post Office Box Q
Philadelphia PA 19105

microtronix

ASCII KEYBOARD TO PET INTERFACE

PRICE $65 Fully Assembled and Checked.

FEATURES:
* Interface any ASCII encoded keyboard to the PET.
* PET keyboard can be used simultaneously.
* No modification to the PET necessary.
* Auxillary power +5V and -12V for your keyboard.
* Jumper selectable for data stobe polarity.
* All PET functions are implemented.
* Recognize all upper case charactors, numeric and punctuations. (NO graphic or lower case)
* One ribbon cable and one connector included.
* 3.5" X 4" single sided PC board.

PET TO S-100 BUS CONVERTER

PRICE $65 Fully Assembled and Checked.

FEATURES:
* Interface to most of the S-100 static RAM and EROM memory boards.
* Interface to most of the I/O Controllers.
* Standard S-100 board size. (4.6" X 10")
* Provisions for memory protect of S-100 boards and system reset.
* Onboard voltage generators for +20 and -20 V DC.
* Power upto 32K bytes of S-100 memory from PET power supply without external power supply.

OPTION #1 - With 6 slot mother board. ADD $15
OPTION #2 - With 6 slot mother board fully stuffed with connectors. ADD $68

TO ORDER

We accept checks, money order, VISA and MASTER CHARGE cards. Personal checks will take 2 to 3 weeks to clear. Please include $2 per item ordered for shipping and handling. Allow upto 4 WEEKS for delivery. SEND FOR FREE SPECIFICATIONS and DETAILS.

EXCEL COMPANY

2241 TAMALPAIS AVE.
EL CERRITO CALIFORNIA
94530

TEL. (415)237-8114

7331

PET Plays MUSIC

Frank Alexander, 570 Juniata Ave., Swarthmore, PA 19081

JR Kinnard showed us how to use the 6522 shift register to make square waves with the PET. Well, once you have an audio oscillator you have the start of a music generator.

The program MUSIC sets up the 6522 as an oscillator, takes a note from DATA statements, interprets it as a setting for repetition-rate command (POKE 59464, R), and holds that note until a timing loop is finished.

You can take a piece of sheet music and write the DATA as note pairs, e.g. C#,2 is a quarter note; F,3 a dotted quarter; Z,0 a short rest; Z,8 a long rest; etc.

All you need for output is a simple loudspeaker & buffer amplifier connected to CB2 on the User Port of the PET. I used an ordinary tape recorder as a handy buffer. These are big signals, about 4 volts. The amplifier protects the PET from inductive kick-back at the ends of the sharp pulses from the 6522.

```
10    POKE 59467,16:REM SET FREE-RUNNING
12    INPUT "D(15,51,81...),T2(100)";D,T3
14    T2=T3
15    POKE 59466,D
17    POKE 59464,0
18    GOSUB 6800
20    READ R$,T
22    PRINT R$"..."T"   "D
25    IF R$="XX" THEN GOTO 900
30    GOSUB 10000
40    GOSUB 6000
90    GOTO 20

499   REM MUSIC NOTES & TIME
500   DATA C1,2,B,2,A,2,G,2,F,2,E,2
 :
 :
582   DATA  C,2,C#,2,D,2,D#,2,E,3,Z,1
584   DATA  C,1,D,1,E,1,F,2,E,2,D,2,C#,2,C,2,C#,2,D,2,Z,1
 :
 :
 :
599   DATA XX,0
900   RESTORE
901   TEST=TEST+1:T2=INT(T2*0.7)
902   ON TEST GOSUB 100,101,102,103,104,105,106,108,109
903   IF TEST> 9 THEN GOTO 14
904   GOTO 15:REM ENDLESS VERSION
908   REM .. RUN 909 TO CLEAR SHIFT REG
909   PRINT PEEK(59467),"59467"
990   POKE 59464,221
995   POKE 59466,0
999   POKE 59467,0:END

6000  POKE 59464,R
6005  IF T< 1 THEN GOTO 6020
6010  ON T GOTO 6100,6200,6300,6400,6600,6800,6900
6020  FOR T1=1 to 2: NEXT T1: RETURN
6100  FOR T1=1 TO T2: NEXT: RETURN
6200  FOR T1=1 TO 2*T2: NEXT: RETURN
6300  FOR T1=1 TO 3*T2:NEXT: RETURN
6400  FOR T1=1 TO 4*T2:NEXT: RETURN
6600  FOR T1=1 TO 6*T2:NEXT: RETURN
6800  FOR T1=1 TO 8*T2:NEXT: RETURN
6900  FOR T1=1 TO 12*T2:NEXT:RETURN
6999  REM TONES,TIME SUB

10000 REM INTERPRETER SECTION
10010 IF R$="B0" THEN R=251
10020 IF R$="C" THEN R=237
10030 IF R$="C#" THEN R=224
10040 IF R$="D" THEN R=211
10050 IF R$="D#" THEN R=199
10060 IF R$="E" THEN R=188
10070 IF R$="F" THEN R=177
10080 IF R$="F#" THEN R=167
10090 IF R$="G" THEN R=157
10100 IF R$="G#" THEN R=149
10110 IF R$="A" THEN R=140
10120 IF R$="A#" THEN R=132
10130 IF R$="B" THEN R= 124
10140 IF R$="C1" THEN R=117
10150 IF R$="C1#" THEN R=111
10160 IF R$="D1" THEN R=104
10170 IF R$="D1#" THEN R= 99
10180 IF R$="E1" THEN R= 93
10190 IF R$="F1" THEN R=88
10200 IF R$="F1#" THEN R=83
10210 IF R$="G1" THEN R=78
10220 IF R$="G1#" THEN R=73
10230 IF R$="A1" THEN R=69
10240 IF R$="Z" THEN R=0
10999 RETURN

100  D=81:RETURN
101  D=85:RETURN
102  D=10:RETURN
103  D=15:RETURN
104  D=20:RETURN
105  D=30:RETURN
106  D=40:RETURN
107  D=50:RETURN
108  D=5:RETURN
109  D=10:RETURN
```

[ed. note: For background for this music program refer to JR Kinnard's article on generating square waves (volume 1, issue 2). Frank's program asks for inputs at line 12 to define the wave and the overall song tempo. Frank's suggested values are 15, 51, or 85 for D, and 100 for T2 (tempo). The program basically loops (lines 20-90) reading the next note and duration pair from DATA statements beginning at 500, and ending when "XX" is read to indicate song end. Lines 901-904 and 100-109 are used to vary the tone and tempo, and could be omitted if you desire.

If you want to pursue Frank's music program, he will send you a cassette with sample songs, instruction sheet, and a schematic for a single transister amplifier for $5.]

PET User Group
PO Box 371
Montgomeryville, PA 18936

FIRST CLASS